REMARKS

Claims 1-36 were presented and examined. In response to the Office Action, no claims are amended, no claims are cancelled and no claims are added. Applicants respectfully request reconsideration of pending claims in view of the above amendments and the following remarks.

I. Allowable Subject Matter

Applicants respectfully acknowledge the Examiner's indication that claims 5-7, 15-17, 22-25, and 27-30 would be allowable if rewritten in independent form including all the limitations of the base claim and any intervening claims.

II. Claims Rejected Under 35 U.S.C. § 103

Claims 1, 11, 21, 26, 31, and 34 are rejected under 35 U.S.C. §103(a) as being unpatentable over *Network Processor Performance Analysis Methodology* by Lakshmanamurthy et al. ("Lakshmanamurthy") in view of *Compilation techniques for parallel systems* by Gupta et al. ("Gupta"). We respectfully traverse this rejection.

Claim 1 recites:

> A method comprising:
> configuring one or more processors into a D-stage processor pipeline;
> **transforming** a **sequential network** application program into **D-pipeline stages** that **collectively perform** an infinite packet processing stage (**PPS**) loop of the sequential network application program; and
> **executing** the D-pipeline **stages** in **parallel** within the D-stage processor pipeline to provide **parallel execution of the infinite PPS loop** of the **sequential network application** program. (Emphasis added.)

While Applicant's argument here is directed to the cited combination of references, it is necessary to first consider their individual teachings, in order to ascertain what combination (if any) could be made from them.

Lakshmanamurthy relates to a network processor performance analysis methodology for networking applications targeted for the IXP 2400 network processor. (See Abstract.) In contrast with Claim 1, Lakshmanamurthy does not disclose or suggest transforming a sequential network application program into D-pipeline stages that collectively perform an infinite packet processing stage (PPS) loop of the sequential network application program, as in Claim 1. Lakshmanamurthy describes a data movement model for estimating compute cycles and total I/O references required for the various operations performed by a network processor on each received packet. Lakshmanamurthy further describes the estimation of the total budget for the packet processing to determine how functional blocks are mapped onto available hardware resources, and how software concepts are used to meet the performance goals. In Lakshmanamurthy, the methodology is validated by implementing microcode and tuning the code (on a simulator and hardware) to demonstrate line-rate performance. (See page 20, left column, lines 10-47.)

Apposite to Claim 1, the disclosure of Lakshmanamurthy is directed to providing a performance analysis of a hypothetical target application if implemented to use the parallel architecture of an IXP 2400 network processor. Lakshmanamurthy provides a line-rate performance based on a data movement model, an estimated number of compute cycles, total I/O references required for operations performed on a packet basis, and a total available budget for packet processing, to enable performance analysis of a hypothetical target application that can be written to run on an IXP 2400 network processor (see *supra*). Nevertheless, Lakshmanamurthy cannot properly be interpreted to disclose transforming a sequential network application program into D-pipeline stages that collectively perform a sequential packet processing stage (PPS) of the sequential network application, as in Claim 1.

As correctly recognized by the Examiner, Lakshmanamurthy fails to disclose or suggest transforming a sequential network application program into D-pipeline stages that collectively perform an infinite packet processing stage (PPS) loop of the sequential network application and executing the D-pipeline stages in parallel within the D-stage processor pipeline to provide parallel execution of the infinite PPS loop of the sequential network application program, as in Claim 1. As a result, the Examiner cites Gupta. However, the Examiner's citing of Gupta fails to rectify the above deficiencies of Lakshmanamurthy.

Gupta relates to compilation techniques for parallel systems (see Abstract). Gupta refers to advances in compilation techniques for uncovering and exploiting parallelism at various granularity levels. Gupta describes compilation techniques for exploiting loop and task level parallelism on shared-memory multiprocessors. (See Abstract.)

In contrast with Claim 1, Gupta does not disclose or suggest transforming a sequential network application program into D-pipeline stages that collectively perform an infinite PPS loop of a sequential network application, as in Claim 1. In Gupta, data dependency analysis is described to determine whether a sequential loop can be transformed into a parallel loop. Neither the sequential nor parallel loops of Gupta collectively perform an infinite PPS loop of a sequential network application. As a result, loop parallelization of Gupta cannot properly be interpreted as teaching transforming a sequential network application program into D-pipeline stages that collectively perform an infinite PPS loop of the sequential network application program, as in Claim 1.

In contrast with the infinite PPS loop of a sequential network application program, as in Claim 1, the compilation techniques described by Gupta merely seek to identify independent code fragments that can execute in parallel. As a result, Gupta cannot rectify the failure of Lakshmanamurthy to disclose or suggest transforming a sequential network application program into D-pipeline stages that collectively perform a sequential packet processing stage (PPS) of the sequential network application program, as in Claim 1.

According to the Examiner, transforming a sequential network application program into D-pipeline stages that collectively perform an infinite PPS loop in the sequential network application program is disclosed by Gupta at section 1, introduction, second paragraph, fourth and fifth paragraphs, section 2, first paragraph, section 2.1, fifth paragraph, section 5.1, fourth paragraph, and section 5.3, second item. The portions of Gupta referred to by the Examiner, however, refer to the desire for identifying code fragments that can be executed in parallel as well as referring parallelizing compilers that automatically restructure programs for execution on parallel architectures. Although referring to various techniques for exploding loops and task level parallelism, as well as describing automatic parallelization compilers, we are unable to discern and the Examiner has failed to identify any portion of Gupta or Lakshmanamurthy that

discloses or suggests transforming a sequential network application program into D-pipeline stages that collectively perform a sequential packet processing stage (PPS) of the sequential network application program, as in Claim 1. We submit that the disclosure of Gupta is merely a wish list and summarization of techniques that could be used for transforming sequential applications into parallel application. However, merely describing a wish list of parallelization techniques, such as automatic parallelization compliers, cannot be properly interpreted as disclosing the transformation of a sequential network application program into D-pipeline stages that collectively perform an infinite PPS loop of the sequential network application program, as in Claim 1.

According to the Examiner, executing the D-pipeline stages in parallel within the D-stage processor pipeline to provide the infinite parallel execution of a PPS loop of a sequential network application program is described in section 1, second paragraph and section 4.1 under loop parallelization of Gupta. Neither the passages referred to by the Examiner nor any other portion of Gupta discloses or suggests executing D-pipeline stages in parallel within a D-stage processor pipeline to provide parallel execution of the infinite PPS loop of the sequential network application program, as in Claim 1. We submit that the current automization compiler techniques described by Gupta are incapable of providing parallel execution of an infinite PPS loop of a sequential network application program since such parallelization is not currently provided by conventional compliers.

Hence, neither the passages referred to by the Examiner, nor any other portions of Lakshmanamurthy in view of Gupta, can disclose or suggest transformation of a sequential network application program into D-pipeline stages that collectively perform an infinite PPS loop of the sequential network application program, as in Claim 1. Furthermore, the automatic parallelization described by Gupta, does not disclose or suggest executing the D-pipeline stages in parallel within the D-stage processor pipeline to provide parallel execution of the infinite PPS loop, as in Claim 1.

Hence, no combination of Lakshmanamurthy and Gupta could disclose, teach, or suggest transforming a sequential network application program into D-pipeline stages that collectively perform an infinite packet processing stage (PPS) loop of the sequential network application,

much less parallel execution of the infinite PPS loop of the sequential network application program, as in Claim 1.

For each of the above reasons, therefore, Claim 1 and all claims which depend from Claim 1 are patentable over the cited art.

Each of Applicant's other independent claims, including Claims 11, 21, 26, 31, and 34, recite features similar to those highlighted above in Claim 1. Therefore, all of Applicant's other independent claims, including Claims 11, 21, 26, 31, and 34, and all claims which depend from them, are also patentable over the cited art for similar reasons. Consequently, please reconsider and withdraw the §103(a) rejection of Claims 1, 11, 21, 26, 31, and 34.

Claims 2, 12, 32, and 35 are rejected under 35 U.S.C. §103(a) as being unpatentable over Lakshmanamurthy in view of Gupta and *Hardware-Software Bipartitioning for Dynamically Reconfigurable Systems* by Rakhmatov et al. ("Rakhmatov"). Also, **Claims 3-4, 8, 10, 13-14, 18, 20, 33, and 36** are rejected under 35 U.S.C. §103(a) as being unpatentable over Lakshmanamurthy in view of Gupta, Rakhmatov, and *Efficient Path Conditions in Dependence Graphs* by Robschink et al. ("Robschink"). In addition, **Claims 9 and 19** are rejected under 35 U.S.C. §103(a) as being unpatentable over Lakshmanamurthy in view of Gupta, Rakhmatov, and Robschink and *A New Approach to the Maximum-Flow Problem* by Goldberg et al. ("Goldberg"). We respectfully traverse these rejections.

DEPENDENT CLAIMS

In view of the above remarks, a specific discussion of the dependent claims is considered to be unnecessary. Therefore, Applicants' silence regarding any dependent claim is not to be interpreted as agreement with, or acquiescence to, the rejection of such claim or as waiving any argument regarding that claim.

Claims 1, 11, 21, 26, 31, and 34 are rejected under 35 U.S.C. §103(a) as being unpatentable over Lakshmanamurthy in view of *NP-Click: A Programming Model for the Intel IXP1200* by Shah et al. ("Shah"). We respectfully traverse this rejection.

Regarding the Examiner's citing of Shah, Shah describes a programming model for the Intel IXP1200 network processor architecture. Shah describes an NP-Click programming model which makes it possible to write efficient code and improve application performance without having to understand all the details of a target architecture. (See Abstract.) According to the Examiner, the collective performance of an infinite PPS loop of a sequential network application program is disclosed by Shah with reference to the Abstract which describes using the programming mode, we implement the data plane of an IPV4 router on a particular network processor. The programming model described by Shah, however, is merely that of a programming model, and as a result, does not disclose the transformation of a sequential network application program into D-pipeline stages that collectively perform an infinite packet processing stage (PPS) loop of a sequential network application program.

According to the Examiner, executing the D-pipeline stages in parallel with the D-stage processor pipeline to provide parallel execution of the infinite PPS loop is disclosed by section 2.2, second paragraph and section 4.42, first paragraph which refer to abstractions used in connection with the IXP1200 architecture. Neither these passages nor any other portion of Shah discloses or suggests executing the D-pipeline stages in parallel within a D-stage processor pipeline to provide parallel execution of the infinite PPS loop of the sequential application program since the disclosure of Shah is merely limited to a programming model and not a technique for transforming a sequential network application program into D-pipeline stages that collectively perform an infinite PPS loop. As a result, the Examiner's citing of Shah fails to rectify the above described deficiencies of the rejection of Claim 1 over Lakshmanamurthy in view of Gupta.

For at least the above reasons, therefore, Claim 1 and all claims which depend on Claim 1, are patentable over the combination of Lakshmanamurthy in view of Shah.

Each of Applicants' other independent claims, including Claims 11, 21, 26, 31, and 34, recite features similar to those highlighted above in Claim 1. Therefore, all of Applicants' other independent claims, including Claims 11, 21, 26, 31, and 34, and all claims which depend from them, are also patentable over the cited art for similar reasons. Consequently, reconsider and withdraw the §103(a) rejection of Claims 1, 11, 21, 26, 31, and 34.

CONCLUSION

In view of the foregoing, it is believed that all claims now pending (1) are in proper form, (2) are neither obvious nor anticipated by the relied upon art of record, and (3) are in condition for allowance. A Notice of Allowance is earnestly solicited at the earliest possible date. If the Examiner believes that a telephone conference would be useful in moving the application forward to allowance, the Examiner is encouraged to contact the undersigned at (310) 207-3800.

If necessary, the Commissioner is hereby authorized in this, concurrent and future replies, to charge payment or credit any overpayment to Deposit Account No. 02-2666 for any additional fees required under 37 C.F.R. §§ 1.16 or 1.17, particularly, extension of time fees.

Respectfully submitted,

BLAKELY, SOKOLOFF, TAYLOR, & ZAFMAN LLP

Dated: September 3, 2009

By: _____
Joseph Lutz, Reg. No. 43,765

1279 Oakmead Parkway
Sunnyvale, CA 94085-4040
Telephone (310) 207-3800
Facsimile (408) 720-8383

CERTIFICATE OF TRANSMISSION

I hereby certify that this correspondence is being submitted electronically via EFS Web to the United States Patent and Trademark Office on September 3, 2009.

_____
Si Vuong